Monsoon 12-Channel Acquisition Board Firmware and Register Definitions

Jamieson Olsen and David Huffman
Fermi National Accelerator Laboratory
Batavia, Illinois, USA 60510

February 16, 2009

Abstract

The 12-channel CCD Acquisition board was designed at Fermilab[1] and will read out large charge coupled devices used in the Dark Energy Survey experiment[2]. The DES hardware is to be installed in the 4m Blanco Telescope located at the Cerro Tololo Inter-American Observatory near La Serena, Chile.

The 12-channel acquisition board is compatible with the Monsoon system developed at the National Optical Astronomy Observatory [3]. The new acquisition board builds upon the existing Monsoon 8-channel acquisition board design[4]. Space limitations imposed by the telescope cage preclude using the 8-channel acquisition boards in this application; hence the 12-channel board was designed to fulfill the tight space requirements.

The first half of this document describes the firmware modules and how they interact with the Monsoon system bus and other components on the 12-channel board. The second half of this document describes the backplane registers, which were designed to be compatible with the 8-channel acquisition board.

Contents

List of Figures

List of Tables

1 Hardware Overview

The primary function of the 12-channel acquisition board (CCDACQ12) is to digitize the analog video signals from the charge coupled devices (CCDs) and send this data over the Monsoon system bus to the master control board (MCB). Secondary functions include generating (and reading back) CCD bias voltages, monitoring temperatures, and storing calibration data. The block diagram of the CCDACQ12 board is shown in Figure 1 and the schematic is available in the DES Document Database [8].

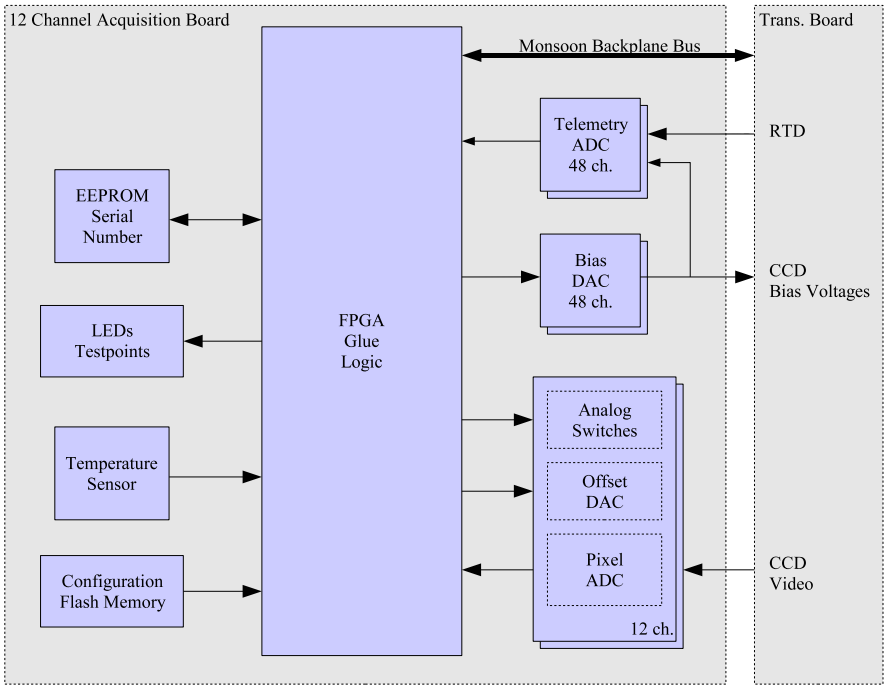

Figure 1: The 12-channel board block diagram.

The CCD video signals are sent through a sensitive analog front end which performs correlated double sampling (CDS) of the waveform. The resulting voltages are digitized with a fast 18-bit analog to digitial converter (ADC) and stored in memory on the acquisition board.

Both 8 and 12-channel acquisition boards are 6U 160mm CompactPCI cards. In the Monsoon system a CompactPCI backplane is used however most of the pin functions have been reassigned. In addition to the analog front end circuits and ADCs the CCDACQ12 board contains 60 independent DAC channels for setting bias and offset voltages and 48 telemetry channels for reading back the bias voltages and external temperature sensors. The CCDACQ12 board power supply requirements are shown in Appendix F.

1.1 8-Channel Acquisition Board Logic

The CCDACQ8 board consists of a central medium-sized field programmable gate array (FPGA) and two complex programmable logic devices (CPLDs) which are essentially used to increase the number of I/O pins. These two CPLDs are connected to the main FPGA through serial buses and this architecture influences the design of the firmware in a few important ways. Commands and data arrive from the backplane in parallel and must be converted to a serial command string in the FPGA. This command string is then transmitted to the proper CPLD, which interprets the command and talks to the target board component (ADC, DAC, etc.). The data collected from the board component is then send back to the FPGA and stored in the internal memory.

1.2 12-Channel Acquisition Board Logic

The CCDACQ12 board uses one large Xilinx [5] FPGA to implement the logic functions. One disadvantage of this approach is that it requires a many I/O pins and it forced us into a 900 "pin" ball grid array (BGA) package. Using one large FPGA does however simplify many things at the firmware level since functional blocks are now inter-connected with parallel buses.

1.2.1 Design Philosophy

Each group of board components (ADCs, DACs, sensors, etc.) are connected to a firmware "wrapper" or interface module that handles the low level communication. In effect, each firmware interface seeks to make each group of board components appear as a small RAM to the rest of the FPGA.

The CCDACQ12 firmware is written completely in VHDL and functionally simulated using Aldec Active-HDL. Models of board components (ADCs, DACs, sensors, etc.) where created and connected to the FPGA in the VHDL testbench code. In the VHDL testbench the user applies stimulus to the Monsoon sequencer bus and can exercise all of the functionality of the board.

2 Firmware Description

The firmware top level block diagram is shown in Figure 2.

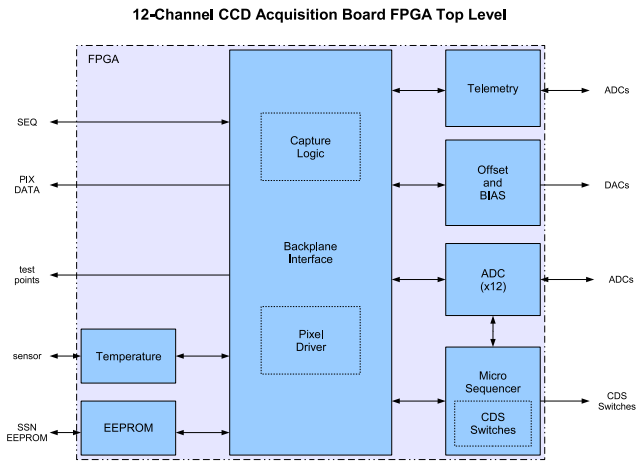

Figure 2: The CCDACQ12 firmware top level block diagram.

2.1 Offset and Bias Module

Bias and Offset voltages on the CCDDAQ12 board are controlled by writing to DACs. There are 48 bias channels and 12 offset channels; all channels are independent and can be changed and read back at any time. Each DAC is 12-bits so values from 0x000 to 0xFFF are allowed.

To the rest of the FPGA this module appears like a RAM 12-bits wide by 60 locations deep. Each memory location corresponds to a physical DAC channel; locations 0-47 map to the bias channels 0-47, and locations 48-59 map to offset channels 0-11. Simply write to the appropriate memory location to update the DAC value. Reading from the RAM will return *the last value written*. Note that RAM writes are synchronous to the master clock but reads are completely asynchronous.

2.1.1 Substrate Voltage Control

Note that bias channels 0 and 1 are used to control the substrate voltage (Vsub) circuit. Bias channel 0 sets the Vsub limit and bias channel 1 controls the slew or ramp rate. The VSUB control bit is in the `CCD_VBIAS` register.

2.1.2 Front End FIFO

Writing to offset and bias DACs is a slow process that takes approximately 3.6 μs per per channel. Certainly the Monsoon sequencer bus — which runs at 40MHz — can supply data faster than it can be written to the DAC chips. For this reason a FIFO buffer has been created in the Offset and Bias Interface module. DAC write requests are buffered in this FIFO and then (slowly) written out to the DAC chips. The FIFO memory used in this module is 64-deep.

The addition of a FIFO buffer enables the Monsoon sequencer bus master to write to the offset and bias DAC registers in a single burst — the user need not worry about losing data because the DAC interface is busy.

2.1.3 DAC Resets

The DAC chips (Analog Devices DAC8420 [7]) can be asynchronously reset to a known value by asserting the CLR_N signal. There are two possible reset values as determined by the state of the CLSEL signal: zero 0x000 (CLSEL=0) or mid-scale 0x800 (CLSEL=1).

Each DAC device contains four channels however some of the bias DAC chips share CLSEL and CLR_N control lines. At the hardware level there are six CLSEL lines and three CLR_N lines shared amongst the 48 bias channels. Likewise, there are three CLSEL lines and three CLR_N lines shared amongst the 12 offset channels. In the FPGA some of these CLR_N and CLSEL lines are tied together to make the backplane register interface consistent with the CCDACQ8 registers; refer to the `CCD_AUXCFGREG` section for more information.

It is important to note that if the CLR_N signal is asserted the DAC output will immediately change to the level determined by the state of the CLSEL line *however the readback value will not change — this value will continue to reflect the last value written to the DAC channel.*

Resetting the CCDACQ12 board will force the CLR_N and CLSEL lines to their inactive states and will not change the DAC outputs, nor will it change the readback values.

2.2 Telemetry Module

The CCDACQ12 board supports 48 telemetry channels which are used to readback the bias voltages and resistive temperature devices (RTDs). Telemetry ADCs are 12-bits and return data in either straight binary to twos complement form depending on how they are configured.

To request that a telemetry ADC begin a new conversion simply write *anything* to the corresponding `CCD_TELEMETRY` register. A short time later the requested data will be available to be read from the same register.

While the telemetry module is busy the user should consider all of the telemetry data to be invalid. The telemetry busy bit is found in the `CCD_STATUSREG` board status register.

2.2.1 Configuring Telemetry Channels

Each telemetry channel has four configuration bits which control how the ADC operates. These bits are: Range (RNG), Bipolar (BIP), and two power down control bits (PD1, PD2). These configuration bits are accessible from the Monsoon sequencer bus interface and may be read or written at any time. Refer to the `CCD_TELMODE` register definitions.

The range and bipolar bits together define the ADC analog input voltage range. These bits should both be set to configure the ADC for -10V to +10V operation. The op-amps and resistor dividers on the CCDACQ12 board have been selected assuming the ADC can accept input voltages in this range. If the ADC is in bipolar mode the data will be returned in twos complement form.

At present time we recommend that the power down control bits (PD1 and PD2) should be cleared, which will leave the telemetry ADC powered up continuously.

When the board powers up these configuration bits will default to all zeros. A board reset will *not* change the values of these bits. These configuration bits are transferred to the ADC only when a conversion is requested.

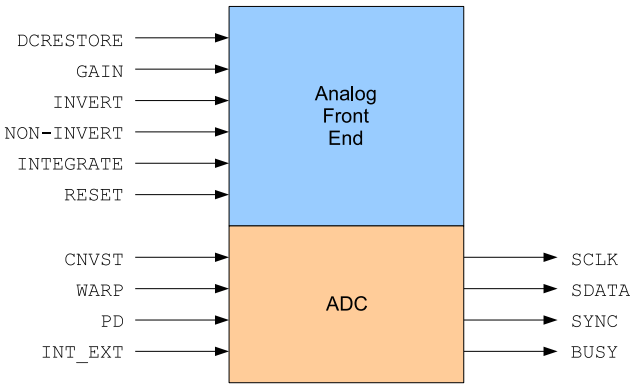

Figure 3: Digital signals connecting to the analog front end circuit.

2.2.2 Front End FIFO

Writing to the telemetry ADCs and waiting for the data is a slow process that takes approximately 21 μs per channel. The telemetry interface module features a 64-deep FIFO to buffer write requests. The addition of the FIFO buffer allows the Monsoon bus master to write to all telemetry channels in one quick burst. It is not necessary to wait for the telemetry module to return to idle to write to the telemetry channels.

2.3 ADC Module

This section describes the operation of the main ADCs on the CCDACQ12 board. There are, coincidentally, twelve main ADCs on the board and these ADCs sample the video signal from the CCDs. Refer to Figure 3 for the block diagram which shows the digital signals which connect the front end circuitry to the FPGA.

There are two ways to tell the ADC to begin a conversion: 1) set the CTC bit in `CCD_CDSREG` register; and 2) use the microsequencer.

The total ADC latency depends on the WARP mode bit. If WARP mode is enabled the conversion and data transmission takes approximately 1.25 μs and the serial data is transmitted *while the ADC is still in conversion*. WARP mode is probably not ideal since the ADCs digital outputs are changing during conversion. For better noise performance we recommend not using WARP mode. If WARP mode is disabled the total latency is approximately 1.75 μs and the serial data is transmitted after the conversion process.

Raw ADC data may be accessed at any time by reading the `CCD_ADCDATA` registers. Normally data from the ADCs is transmitted automatically on the pixel data bus if *pipeline mode* is enabled in the `CCD_CCDCTRL` board control register.

2.3.1 ADC Clocking Scheme

Data from the ADC is transmitted serially back to the FPGA synchronous to the ADC clock and thus there are twelve ADC clock inputs into the FPGA. The problem is that the Spartan III FPGA has only eight dedicated clock inputs. One solution to this problem is to route some of the ADC clocks on non-global resources in the FPGA. While on the surface this might seem acceptable it should be noted that skew on non-global clock nets in FPGAs is notoriously difficult to control and can lead to race conditions and garbled data.

Another solution is to use a fast local clock to over-sample the ADC clock inputs and look for the ADC clock edges. The CCDACQ12 uses the local 40MHz oscillator and an internal digital clock manager (DCM) to create a 160MHz clock in the FPGA which is used for this purpose. Note that this 160MHz clock net is completely internal to the FPGA. The DCM's "locked" status bit is available in the `CCD_STATUSREG` register. If for any reason the DCM is reporting an unlocked condition it must be manually reset by writing to the the `CCD_CCDCTRL` board control register.

2.3.2 Physical and Logical Channels

The term "physical channel" refers to the actual ADC component on the CCDACQ12 board. There are 12 physical channels numbered from 0 to 11.

The CCDACQ8 board introduced the concept of "logical channels" as a way to conveniently group physical channels together for certain operations such as setting CDS switches or triggering ADC readouts. Both acquisition boards support eight logical channels numbered from 0 to 7. Each physical channel maps to a logical channel. The `CCD_CDSREG` and `CCD_SEQTRIG` registers reference *logical channel* numbers. The mapping between physical and logical channels is specified in the `CCD_ADCCFG` register block.

2.3.3 ADC Configuration

Each ADC physical channel has a corresponding register in the `CCD_ADCCFG` register block. This configuration register is used to control the ADC power and WARP modes, the CDS gain switch, the logical channel mapping, and the test modes. Upon powerup these configuration bits are all zero. A board reset will force these bits to zero as well.

Disabling the low power standby mode is recommended because the ADC may exhibit some strange non-linear behavior as it warms up to operating temperature.

The test mode bits select between normal data (00), which is the power on reset default, and the following test modes: force all zeros (01), count-up (10), and pseudo-random (11). In count-up mode the pixel values are (in decial) xxyyzz where xx=board priority, yy=physical channel number, and zz is a modulo-100 counter.

When any of the test modes are enabled no "convert start" pulse is sent to the physical ADC device. The counter mode begins from zero and counts up. The pseudo-random number generator is based on a 32-bit linear feedback shift register (LFSR) with a period of approximately 4 billion values. Each physical channel uses a different seed value for the LFSR logic.

2.4 Temperature Sensor Module

Ambient temperature is monitored using the AD7814 device from Analog Devices. The temperature sensor module handles all communication with the sensor and converts the serial data stream into a parallel format that can be read from the backplane.

Note that the temperature sensor is not read automatically. The user must first write to the `CCD_TEMP` register to force an update. While the temperature sensor interface is communicating with the sensor it will assert the "TEMP BUSY" bit in the `CCD_STATUSREG` register. While the temperature sensor module is busy do not attempt to read the `CCD_TEMP` register.

2.5 Serial Number and Non-Volatile Memory

Each board contains a "one wire" serial EEPROM device with a unique serial number. This interesting little chip is powered from a single bidirectional data signal. The EEPROM interface module handles all of the handshaking with the device and performs a CRC check to insure that the data is valid. The 48bit serial number data can be accessed by reading the `CCD_SERNUM` register.

Reading the serial number and EEPROM contents requires approximately 60ms and occurs automatically upon powerup and after a a hard or soft reset. A pair of status bits ("OWD busy" and "S/N CRC") may be accessed by reading the `CCD_STATUSREG` register.

The small "one wire" device also contains 4k bits of non-volatile EEPROM memory which is available to store calibration data. Upon power up the EEPROM data is copied into a RAM buffer which maps to the `CCD_NVDAT` block of registers. This memory is organized as 256 16-bit words and may be read or written at any time. If changes are made to the data it must be saved back to the EEPROM by writing to the `CCD_NVCTL` register. Writing back to the EEPROM is a relatively slow process (200ms) during which time writes to `CCD_NVDAT` will be ignored.

2.6 Test Points and LED Driver Module

The CCDACQ12 board has four LEDs: one LED on the front panel and three others near the backplane connector. All of these LEDs may be forced on by writing to the `CCD_LEDCTL` register. The front panel

LED normally blinks to indicate that the board registers are being accessed from the sequencer bus. All LEDs are pulse stretched with with monostables so that extremely fast pulses are visible.

With the exception of EXT1 and EXT2 the front panel test points are left undefined. This is done to allow "probes" to be added on an as-needed basis for firmware debugging. Refer to Appendix D for a pinout of the front panel connector.

2.7 Backplane Interface Module

The backplane interface module is the "glue logic" that connects the various interface blocks to the Monsoon backplane bus. It is in this module that the sequencer bus input signals are registered, register addresses are decoded, and a large output mux is implemented to drive the pixel data bus.

Due to tight timing on the pixel data bus all backplane reads are completely asynchronous and thus do not require any clocks. Backplane writes to the CCDACQ12 board are synchronous to the master board clock.

2.7.1 Pixel Driver Module

Once the serial data is received from the ADCs the FPGA updates the `CCD_ADCDATA` register block so that the ADC data may be read at any time. The `CCD_ADCDATA` registers are quite useful for debugging purposes when easy access to the latest ADC data is required. During data acquisition however the additional overhead associated with reading the individual `CCD_ADCDATA` registers has a negative impact on system performance. Direct memory access (DMA) modes enable the CCDACQ12 board to automatically transfer the ADC data across the Monsoon backplane to the host. Two DMA modes are defined:

Pipeline Mode When pipeline mode is enabled the FPGA first waits until the ADC data has been received. Then the FPGA checks the status of the PIPE_REQ bus to see if any other boards are driving the pixel bus. At the appropriate time the FPGA takes control of the pixel data bus, dumps out the ADC data, and finally releases the bus.

Burst Mode Burst mode is similar to pipeline mode, however the process of checking the PIPE_REQ bus does not begin until the user writes to the `CCD_BURST` register.

2.8 Microsequencer Module

By far the most common board operations involve toggling CDS switches and reading out the ADCs. Normally the MCB would perform these tasks by repeatedly writing over the backplane to the `CCD_CDSREG` register. Since the steps needed to acquire and read out a single pixel are highly repetitive it makes sense to provide a mechanism to do this locally on the acquisition board. The CCDACQ12 microsequencer enables the user to define many CDS switch states and step though these states at precise time intervals. The microsequencer logic consists a RAM buffer, control/status register, and a small finite state machine.

The microsequencer pattern memory `CCD_PATTMEM` is where the user specifies the CDS switch states and delay values. The pattern memory buffer is RAM based and may be read at any time. Writes to the pattern memory will be ignored if the microsequencer is busy. The pattern memory block consists of up to 64 CDS switch states or vectors. Each vector specifies a delay value (in steps of 25 ns) followed by two external trigger bits, five CDS switch values and the ADC CTC control bit. The external trigger bits are front panel digital outputs used primarily for debugging.

Writes to the `CCD_SEQTRIG` register will trigger the microsequencer. When writing to this register the user must specify the address of the starting vector, the number of vectors to process, and the applicable logical channel(s).

The microsequencer finite state machine implements a simple and straightforward algorithm: 1) assert the control bits, 2) wait for the specified delay, 3) fetch the next vector and repeat until the specified number of vectors have been processed. Note that the minimum delay value is 25 ns and the maximum delay value 6.4 μs.

2.8.1 CDS Switch Module

The CDS switches control the operation of the analog front end circuit and are used to condition the video signal prior to digitization. The CDS switches are: Reset, DC Restore, Integrate, Invert, Non-Invert and Gain. The Gain switch is not normally changed during data acquisition and is located in the `CCD_ADCCFG` register.

The CDS switch module implements the `CCD_CDSREG` register. It may be written directly from the backplane or indirectly through the microsequencer. Reading the `CCD_CDSREG` will return the last value written to this module.

When data is written to the CDS switch module *logical* channels are referenced; thus this module must map the logical channels onto the physical channels. The mapping between logical and physical channels occurs in the `CCD_ADCCFG` registers.

Note that with the exception of the gain switch, all CDS signals are inverted in the FPGA to compensate for the normally closed analog switch chips.

2.9 Firmware Simulation

The CCDACQ12 firmware was completely written in VHDL. Where appropriate Xilinx primitives are explicitly declared and instantiated in the code. The firmware has been functionally tested using the Aldec Active-HDL development tools. A full timing simulation of the firmware has not been performed — rather, the design was carefully constrained and the Xilinx static timing analyzer was relied upon to verify that the design meets all timing constraints.

A complete model of the acquisition board was created in the simulator environment. Functional models of board level components (ADCs, DACs, etc.) were created in VHDL and connected to the FPGA. It is possible to simulate most of the acquisition board and observe interactions between the FPGA, DACs, ADCs, and other board sensors.

2.9.1 Testbench

A testbench is special VHDL file which is used by the simulator to apply stimulus to the device under test — in this case the device is a VHDL file which instantiates the FPGA and other board component models. The testbench drives the Monsoon sequencer bus and can read or write to any of the registers in the FPGA.

There are essentially two ways to drive the sequencer bus: the first approach involves manipulating the data and control signals explicitly at the bit level, and the second approach describes backplane operations at a higher level (e.g. "write XXXX to register YYYY then wait for ZZZZ"). This testbench uses the second approach and implements three functions to define three operations on the sequencer bus: `bp_write16()`, `bp_write32()`, and `bp_read()`. The following VHDL fragment shows a backplane write followed by a delay and a backplane read:

```
bp_write32(CCD_CDSREG,0xDEADBEEF);
wait for 100ns;
bp_read(CCD_STATUSREG);
```

The `bp_read()` function prints to the console the value read from the data bus – so it is entirely possible to test the design without having to study any waveforms.

Loops and other conditional control structures can be defined in the testbench as well. In the following example a loop is used to write to all bias DACs:

```
biasloop: for i in 0 to 47 loop
  bp_write16(CCD_VBIAS+i,0x123);
  wait for 100ns;
end loop biasloop;
```

The register names are mapped to physical backplane memory addresses in the package file `ccdacq_package.vhd`. Since registers are referenced by name throughout the design the package file must be included with the rest of the source during synthesis.

2.9.2 Board Component Models

In order to simulate the interaction between the FPGA the rest of the board it was necessary to write VHDL models for a few board components. These following models were created based on the timing diagrams in their respective datasheets:

- Temperature sensor (AD7814)
- "One Wire" EEPROM and Serial Number (DS2433) [6]
- Single channel ADC (AD7674)
- Eight channel ADC (MAX1270)
- Quad DAC (DAC8420)

Since these components models were never intended for synthesis all of the features in VHDL are used: delays, integers, floating point numbers, etc. For example the MAX1270 ADC model has eight floating point inputs which are "sampled" and converted into a binary number which is shifted out one bit at a time back to the FPGA.

3 The Monsoon Backplane Bus Interface

The Monsoon sequencer bus consists of the following signals driven by the MCB: clock, board select (SEL_N), mode (SEQ_MODE[1:0]), a device address bus (DEV_ADDR[2:0]) and a shared address/data bus (SEQ_DATA[31:0]). All of these signals are synchronous to the 40MHz clock and are valid on the *rising edge* of the clock. Table 1 defines the four types of bus operations.

Table 1: Sequencer Bus Modes

SEQ_MODE	Operation	Address	Data
00	Reset	n/a	n/a
01	Read	SEQ_DATA[31:16]	n/a
10	16-bit Write	SEQ_DATA[31:16]	SEQ_DATA[31:16]
11	32-bit Write	DEV_ADDR[2:0]	SEQ_DATA[31:0]

For 32-bit write operations all of the bits in the SEQ_DATA bus are used for data and the target address is specified using the three bit DEV_ADDR bus. This means that there are only eight 32-bit registers available, and in fact the acquisition board only uses four 32-bit registers. The DEV_ADDR bits are not used for read and 16-bit write operations.

Read operations and 16-bit write operations specify the target address in the upper 16-bits of the SEQ_DATA bus. For 16-bit write operations the data word is specified in the lower half of SEQ_DATA. Writes to addresses 0x0000-0x0007 are not valid when writing in 16-bit mode. Read operations return the requested data on the 48-bit pixel data bus.

An active low acknowledge signal (ACK_N) is asserted during read and write cycles, refer figures 5 and 4 for details.

3.1 Resetting the Board

The acquisition board defines two types of reset operations: *hard reset* and *soft reset*. A hard reset (sometimes called "reboot") completely erases the FPGA and reloads the firmware from the onboard FLASH PROM. After a hard reset all internal logic, registers and memories return to their default state. A soft reset forces all internal logic to a known state and clears most registers while the memory contents (`CCD_PATT_MEM`, `CCD_TELMODE`, etc.) are unaffected. There are three ways to soft reset the acquisition board:

- Push the reset button on front panel.
- Write to the `CCD_IDENTREG` register.

- Set DEV_ADDR='00000' when SEQ_MODE='00'.

and there are three ways to hard reset the acquisition board:

- Cycle power.
- Write to the `CCD_FIRMVERS` register.
- Set DEV_ADDR='11111' when SEQ_MODE='00'.

After a hard reset the acquisition board will be unresponsive for approximately 30 μs and some garbage may appear on on the pixel data bus.

3.2 Backplane Capture Buffer

The firmware has the ability to store up to 1024 writes to the board and read them back for debugging. The address, write type (16-bit or 32-bit), and data are stored in a static ram buffer in the FPGA. Resetting the board has no effect on the contents of this capture buffer. Rebooting the board will however erase the buffer contents. For more information see Section 4.16.

3.3 Bus Timing

3.3.1 Write Cycles

The Monsoon backplane supports 32-bit and 16-bit write cycles; both types are synchronous to the master clock. The CCDACQ12 board requires a minimum setup time (t_{SU}) of 8ns and a minimum hold (t_H) time of 5ns with respect to the backplane clock. The CCDACQ12 board does not require any wait states between write cycles.

The ACK line is normally tri-stated but is pulled low during write transactions as shown in figure 4. The ACK line is driven by a registered copy of the board select (SEL_N) signal and has a clock-to-out delay (t_{ACK}) of approximately 8ns.

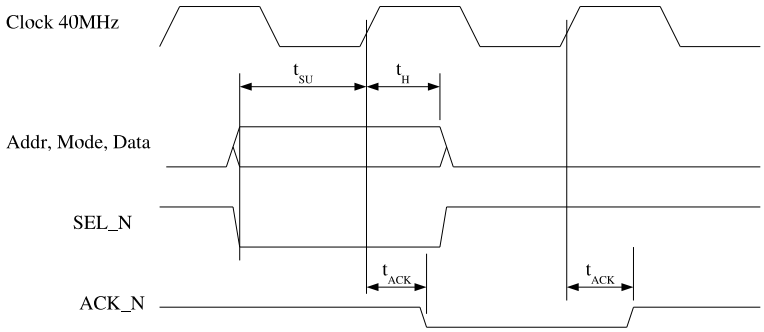

Figure 4: Monsoon Sequencer Bus Write Cycle Timing

3.3.2 Read Cycles

All read transactions are asynchronous in nature. The logic which drives the pixel data bus is purely combinatorial and depends only upon the state of the board select, sequencer mode and address lines. The decision to use combinatorial logic was driven by a tight read cycle latency (t_{ARL}) constraint of 25ns.

3.3.3 Pipeline and Burst Mode Data Transfers

The acquisition board has the ability to transfer pixel data across the PIX_DATA bus without the intervention of the MCB. Automatic data transfers are enabled whenever the *PIPELINE MODE* or *BURST MODE* control bits are set in the `CCD_CTLREG` control register.

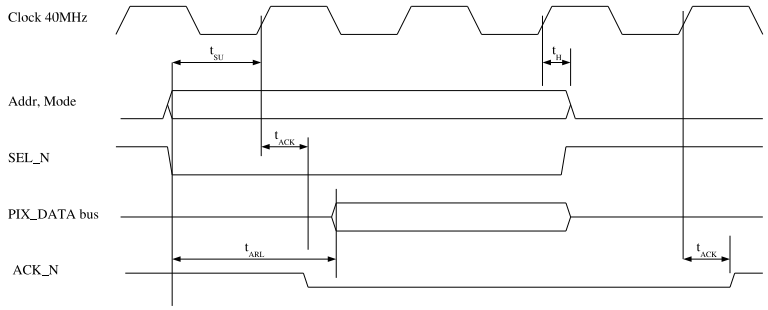

Figure 5: Monsoon Sequencer Bus Read Cycle Timing

Since several acquisition boards share the backplane it is necessary to control access to the pixel data bus so that bus contention does not occur. The bus arbitration scheme implemented here is priority based — each acquisition board is assigned a priority value between zero (highest priority) and six (lowest priority). When an acquisition board has data ready to send it first announces that it has data to send, then waits for some time for the bus to become available. If the pixel data bus is available the acquisition board takes control of the bus and sends the pixel data to the MCB. Pixel data is transferred two 18-bit data words per clock cycle for a maximum throughput of 80M pixels per second. The "low pixel" maps to PIX_DATA[17:0] and the "high pixel" maps to PIX_DATA[41:24]. Unused pixel data bits are forced to zero.

An 8-bit bidirectional control bus called PIPE_REQ_N is used by the acquisition boards to determine who has control of the pixel data bus. All acquisition boards drive the PIPE_REQ_N bus with open collector drivers and pullup resistors are located on the MCB. The most significant bit on the PIPE_REQ_N bus is asserted (pulled low) by the acquisition board when driving the pixel data bus. The lower seven bits of the PIPE_REQ_N bus are used to identify the priority level of the board driving the bus.

Figure 6 illustrates how two acquisition boards share the PIX_DATA bus in pipeline mode. In this example both boards get their pixels ready at the same time. The highest priority board (0) waits briefly, then transmits two pixels, then releases its priority line. The second board (priority=1) then detects that no higher priority boards have the bus and then it takes control of the bus and transmits 2 pixels and releases the bus. Each board pulls low the PIPE_REQ_N[7] line when driving the pix data bus and the MCB latches the pixel values on the RISING EDGE of the backplane clock. Note that the ACK_N bus signal is *not asserted* during pipeline write operations.

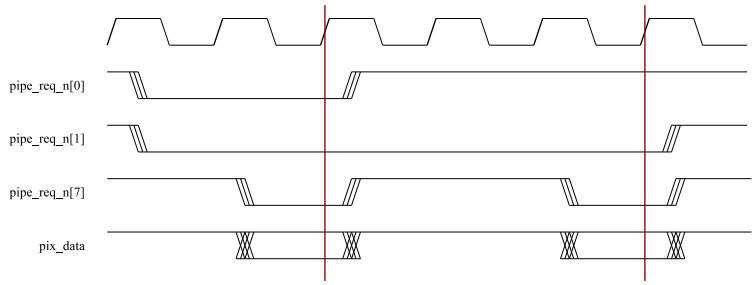

Figure 6: An example pipeline mode write cycle.

Pipeline and Burst modes work very much the same way with one exception: pipeline mode automatically transmits the data while burst mode waits for the user to write to the `CCD_BURST` register.

4 Backplane Registers

4.1 Correlated Double Sample Register (CCD_CDSREG)

The `CCD_CDSREG` register directly controls the CDS switches and the ADC CTC ("start conversion") control bit. This register may be read or written at any time. The `CCD_CDSREG` register is mapped into

the 32-bit address space but only the lower 16 bits are defined as shown in figure 7.

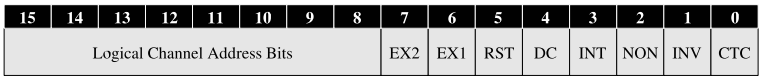

Figure 7: The Correlated Double Sample Register

The lower byte of the `CCD_CDSREG` register directly controls the external trigger bits (EX2 and EX1), CDS reset (RST), DC Restore (DC), Integrate (INT), Non-Invert (NON), Invert (INV), and the ADC CTC control bit. Asserting the CTC bit will begin an ADC conversion; this bit is automatically cleared to prevent multiple conversions.

The upper byte of the `CCD_CDSREG` register specifies which *logical channels* should be updated. The lower seven bits of this byte are bitmapped: setting bit zero of this byte selects logical channel zero, bit one selects logical channel 1, and so on. Setting the most significant bit of this byte selects all channels. The mapping between physical and logical channels occurs in the `CCD_ADCCFG` registers.

For example, writing 0x4393 to the `CCD_CDSREG` will turn on the front panel external trigger output 2, it will turn on the DC Restore and Invert switches and begin an ADC conversion. Only physical channels which have been assigned to logical channel 2 will be affected.

Reading this register will return the last value written to it. Note that when triggered the microsequencer will write to this control register.

4.2 Digital Output Port Register (CCD_DOPREG)

The data written to this 32-bit register has no impact on the board operation whatsoever. Contrary to it's name the `CCD_DOPREG` is not connected to any outputs or test points on the board. The primary purpose of the `CCD_DOPREG` register is to test backplane reads and writes. A board reset will force the contents of this register to zero.

4.3 Burst Command Mode Register (CCD_BURST)

Write anything to this 32-bit register to initiate a transfer of ADC data to the MCB. The number of pixels that will be transferred is specified in the `CCD_XFERCOUNT` register and the physical channel numbers are specified in the `CCD_REDIRECT` register block. See the `CCD_REDIRECT` section for details.

The burst mode bit must be set in the `CCD_CTLREG` board control register otherwise write to the `CCD_BURST` register will be ignored.

Reading the `CCD_BURST` register is an undefined operation and will most likely return all 1's.

4.4 Sequencer Trigger Register (CCD_SEQTRIG)

Figure 8: The Microsequencer Trigger Register `CCD_SEQTRIG`

Write to this 32-bit register to trigger the microsequencer. When writing to this register the user must specify three arguments: the logical channel address bits, the address of the first vector, and the number of vectors to process. Refer to figure 8 for bit positions of the logical channel address, run length, and starting address fields.

Reading the `CCD_SEQTRIG` register will return the value of the microsequencer address pointer in the starting address field; all other bits will be zero.

4.5 ADC Raw Data Registers (CCD_ADCDATA)

After each ADC conversion the raw data will be placed in this block of registers. There are twelve registers which correspond to the twelve physical channels on the board. Each register contains an 18-bit number and may be read at any time.

These registers are primarily used for debugging. Normally the ADC data will be sent back to the MCB "automatically" using *pipeline* or *burst* transfer modes.

4.6 ADC Configuration Registers (CCD_ADCCFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	Test Mode		X	X	X	Logical Channel			X	CDS GAIN	WARP	PWR DWN

Figure 9: The ADC Configuration Registers `CCD_ADCCFG`

This block of twelve registers configures the main ADCs. Each physical channel numbered zero though eleven has a corresponding register in this block.

The default state of each `CCD_ADCCFG` register is all zeros.

The ADC may be placed into a power down mode by setting the PWR DWN bit. Note that the ADC will power down after the next conversion.

Warp mode is used to reduce the acquisition time by reading out the ADC during conversion. Using warp mode will likely increase noise and is not recommended for data taking.

Setting the CDS GAIN bit fixes the gain of the front end circuit at 2x while clearing the CDS GAIN bit results in unity gain.

Each physical channel maps to one logical channel address which is specified in bits [6:4]. Valid logical channel addresses are zero through six; setting the logical channel address to seven ('111') is equivalent to setting it to zero.

The test mode bits select between normal data (00), all zeros (01), counter (10) and pseudo-random data (11). In counter mode the pixel values returned are in the form xxyyzz (decimal) where xx=board priority, yy=physical channel, and zz is a modulo-100 counter. This mode is useful for uniquely identifying where the pixels came from when using an image viewer like DS9 which displays pixel values in decimal form.

Note that on the CCDACQ8 board setting bit 9 changes the value read back from each `CCD_ADCCFG` register. This functionality has not been implemented on the CCDACQ12 board firmware.

4.7 Offset DAC Registers (CCD_VOFFSETS)

The acquisition board has twelve offset voltages which are set with DACs. Each 12-bit DAC is independent and maps to a single register in the `CCD_VOFFSETS` block. Writing to one of the `CCD_VOFFSETS` registers will set the DAC; reading from the register will return the last value written.

The offset DACs are also controlled by two bits in the `CCD_AUXCFGREG` control register. All offset DAC channels may be forced to zero or mid-scale by asserting the OFFS CLR and OFFS CLSEL bits — see section 2.1.3 for details.

The offset DACs and `CCD_VOFFSETS` registers are unaffected by soft resets; a hard reset will force these registers to zero.

4.8 Bias DAC Registers (CCD_HVBIASES)

The acquisition board supports up to 48 bias voltages which are set with DACs. Each 12-bit DAC is independent and maps to a single register in the `CCD_HVBIASES` block. Writing to one of the `CCD_HVBIASES` registers will set the DAC; reading from the register will return the last value written.

The bias DACs are also controlled by four bits in the `CCD_AUXCFGREG` control register. All bias DAC channels may be reset asserting the BIAS CLR control bit. The CLSEL bit determines if the DAC will be reset to zero (CLSEL='0') or mid-scale (CLSEL='1'). The 48 bias DAC channels are divided into six CLSEL groups, see table 2 for details. In order to be somewhat consistent with the

Table 2: Bias DAC CLSEL Groups

CLSEL line	Bias Channels
0	0-7
1	8-15
2	16-23
3	24-31
4	32-39
5	40-47

CCDACQ8 board bias CLSEL groups 0&2, 1&3, and 4&5 are merged in the `CCD_AUXCFGREG` control register.

Refer to section 2.1.3 for more information on the CLSEL and CLR DAC control bits.

Bias DAC channels 0 and 1 are used to control the substrate voltage (Vsub) circuit. Bias channel 1 sets the slew rate and bias channel 0 sets the upper limit voltage. The Vsub circuit is turned on and off in the `CCD_VSUB` control register.

The bias DACs and `CCD_HVBIASES` registers are not affected by soft resets; a hard reset (or reboot) will force these registers to zero.

4.9 Telemetry Mode Registers (CCD_TELMODE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	X	X	X	X	BIP	RNG	PD1	PD0

Figure 10: The telemetry configuration register `CCD_TELMODE`.

Each telemetry ADC channel is independent and has four configuration bits as shown in figure 10. Each register in the `CCD_TELMODE` register block corresponds to a single telemetry ADC channel. Refer to secton 2.2.1 for more detail on how the four configuration bits work.

The configuration bits will be applied to the physical ADC channel upon the next conversion cycle which is initiated by writing to the `CCD_TELEMETRY` registers.

A hard or soft reset will force the `CCD_TELMODE` registers to zero.

4.10 Telemetry Registers (CCD_TELEMETRY)

The acquisition board implements 48 telemetry channels for reading back bias voltages and RTDs. Each telemetry channel consists of a 12-bit ADC and is configured in the `CCD_TELMODE` registers.

Write anything to a register in the `CCD_TELEMETRY` block and the corresponding telemetry ADC will begin its conversion cycle. While any of the telemetry ADCs are busy the "TEL BUSY" bit will be set in the `CCD_STATREG` status register. The RAW 12-bit ADC values may be accessed by simply reading the `CCD_TELEMETRY` registers. Note that the `CCD_TELEMETRY` registers should not be read while the "TEL BUSY" bit is set.

Writes to the `CCD_TELEMETRY` register may occur even if the "TEL BUSY" bit is set since the telemetry front end firmware incorporates a FIFO buffer. Refer to section 2.2.2 for details.

4.11 Auxiliary Control Register (CCD_AUXCFGREG)

The `CCD_AUXCFGREG` control register is shown in figure 11. This register is readable and writable at any time; a hard or soft reset will force this register to zero.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	BIAS 4,5	OFFS CLSEL	BIAS 1,3	BIAS 0,2	OFFS CLR	BIAS CLR	X	X	BIAS EN	CDS EN

Figure 11: The Auxiliary Board Configuration Register `CCD_AUXCFGREG`

The CDS switches are enabled by asserting the "CDS EN" control bit. If the CDS switches are disabled the signals will be tri-stated; it is not known how the analog front end circuit will operate when the CDS control signals are floating.

All bias voltages are enabled by asserting the "BIAS EN" control bit. This control bit turns on the CMOS switches which connect the bias DACs to the backplane transition board. The bias voltages are always connected to the telemetry ADCs, however.

The remaining control bits in the `CCD_AUXCFGREG` register are used to reset the offset and bias DACs. Refer to sections 4.8 and 4.7 for more information on how these control bits work.

4.12 Channel Redirection Registers (CCD_REDIRECT)

When the acquisition board is configured for *burst* or *pipeline* transfer mode the `CCD_REDIRECT` register block specifies which ADC channels are to be sent to the MCB. The `CCD_REDIRECT` block consists of 24 4-bit registers; each register references a *physical channel* number. For example, suppose the registers are setup like this:

```
CCD_REDIRECT(0) = 4
CCD_REDIRECT(1) = 5
CCD_REDIRECT(2) = 3
CCD_REDIRECT(3) = 1
CCD_XFERCOUNT   = 4
```

When a burst or pipeline transfer occurs four pixel values will be sent to the MCB. Two pixel values are sent in each backplane clock. In the first clock cycle channel 4 will be sent in the lower half of the pixel bus and channel 5 will be sent in the upper half of the pixel bus. In the following cycle channels 3 and 1 will be sent to the MCB. Refer to section 3.3.3 for more details on backplane timing and control in pipeline and burst modes.

The acquisition board does *not* check that the physical channels specified in the `CCD_REDIRECT` registers have *new data*. These registers default to zero and are NOT affected by reset.

4.13 Transfer Count Register (CCD_XFERCOUNT)

When the acquisition board is configured for *burst* or *pipeline* mode the number of pixels transferred is set by the `CCD_XFERCOUNT` register. Valid xfercount values are 2, 4, ... 24. Single pixel "blogger mode" is no longer supported in the firmware.

4.14 Microsequencer Pattern Memory (CCD_PAT_MEM)

The `CCD_PAT_MEM` registers are implemented as a 16x64 RAM block. Each memory location in this buffer contains a 16-bit vector as shown in Figure 12.

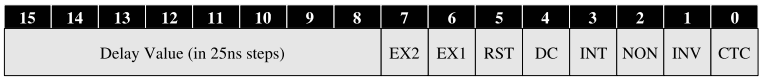

Figure 12: The microsequencer pattern memory vector format.

Each vector specifies a delay value (in steps of 25ns) followed by the state of the CDS switches and the ADC CTC control bit.

The `CCD_PAT_MEM` buffer is unaffected by a soft reset but a hard reset will force all 64 vectors to their default values. The `CCD_PAT_MEM` memory may be read at any time but writes should occur only when the microsequencer is idle. Refer to section 2.8 for more information on the operation of the microsequencer.

4.15 Non-Volatile Memory (CCD_NVDAT)

This block of memory used to store calibration constants or other data that must stored in non-volatile memory. Upon powerup the FPGA reads 4k bits from the DS2433 serial EEPROM device and stores

the data in the CCD_NVDAT buffer. This buffer is 256 16-bit words and may be read or written at any time.

The user *must* write to the CCD_NVCTL control register to save the data back into the DS2433's EEPROM memory, see below.

4.15.1 Non-Volatile EEPROM Control Register (CCD_NVCTL)

To save the contents of the CCD_NVDAT buffer to the EEPROM write 0x1234 to the the CCD_NVCTL control register. *The save operation does not happen automatically; it is up to the user to insure that the data is saved back to EEPROM!* Writing to the EEPROM is a slow process that requires approximately 200ms. While the EEPROM is busy writes to CCD_NVDAT will be ignored and the "OWD BUSY" (one wire device busy) bit will be set in the board status register CCD_STATREG.

4.16 Backplane Capture Buffer

Maintaining a history of the commands written to the board can be a useful debugging tool. Logic contained in the FPGA saves up to 1024 backplane writes to the board. The command buffer is random access controlled through a 10-bit pointer in the CCD_CAPCTRL register.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR	W16	W32				PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

Figure 13: The Backplane Capture Control Register.

The command buffer must be explicity cleared by setting the most significant bit in the CCD_CAPCTRL register shown in Figure 13. After the buffer is cleared the next 1024 writes to the board are stored (writes to the CCD_CAPCTRL register are not stored, though). Once the buffer fills up with 1024 commands it stops storing commands.

At any time the 10-bit pointer can be written and the command address, data, and flags can be read back through the CCD_CAPADDR, CCD_CAPDAT0 and CCD_CAPDAT1 registers. Both 32-bit and 16-bit commands are stored in the buffer; to determine if the stored command was 16 or 32 bit check the W16 and W32 flags in the CCD_CAPCTRL register. An example:

- Write 0x8000 to CCD_CAPCTRL to clear the command buffer.
- Write a bunch of commands to the board.
- Write 0x0000 to CCD_CAPCTRL. This sets the pointer to the beginning of the buffer.
- Read the CCD_CAPCTRL and note the value of the 16 and 32 bit write flags.
- Read the CCD_CAPADDR register. This first stored command was written to this address.
- Read the two data registers CCD_CAPDAT0 and CCD_CAPDAT1 to see what data value was written.
- Increment the pointer to the next command in the buffer and loop until all commands are read back.

4.17 Acquisition Audit Registers

The acquisition audit registers are used to log significant events that occur to produce pixel data. These three registers are cleared after a reset or a write to the CCD_EVENT_REG occurs.

CCD_CTC_COUNT This 16-bit register counts the number of times the ADC CTC bit is asserted. The CTC bit, which initiates an ADC conversion cycle, may be asserted either directly by writing to the CCD_CDSREG control register or indirectly by firing the microsequencer.

CCD_REQ_COUNT This 16-bit register counts the number of times a pipeline data transfer cycle occurs. Under normal conditions the number of CTC strobes will equal the number of data transfer cycles. However if WARP mode is enabled the ADCs will require an extra CTC strobe to "warm up" prior to pixel data acquisition and other CTC strobe to get the last pixel value out at the end.

CCD_XFER_COUNT This 16-bit register counts the actual number of pixels sent.

4.18 Global Event Register (CCD_EVENT_REG)

The `CCD_EVENT_REG` register is write only and is used to clear the audit registers if the most significant bit (15) is set. This operation is called "Start Exposure" in the CCDACQ 8 channel documentation.

4.19 Substrate Voltage Control Register (CCD_VSUB)

This register is readable and writable and controls the analog circuits that produce the CCD substrate voltage. To turn on the substrate voltage set the least significant bit of this register. To turn off the substrate voltage clear the least significant bit of this register. Note that the substrate voltage ramps up and down at a rate proportional to the voltage on bias channel 1, and the upper limit of the substrate voltage is set by the voltage on bias channel 0.

4.20 Front Panel LED Control Register (CCD_LEDCTL)

This register is readable and writable and directly controls the four LEDs on the board. The three least significant bits [2:0] drives the three user LEDs near the backplane connectors. Setting bit three overrides the front panel "DTACK" LED. A hard reset forces this register to zero.

4.21 Silicon Serial Number Register (CCD_SERNUM)

The 48-bit board serial number may be accessed by reading the `CCD_SERNUM` register. This resister is read-only; writes to this register will be ignored. Immediately after power up (or reset) the firmware will read the 48-bit serial number from the "one wire" EEPROM device and perform a CRC check. This process takes approximately 60ms during which time the "EEPROM BUSY" bit is set in the `CCD_STATREG` register. If the CRC check passes the "S/N CRC" bit is set in the `CCD_STATREG` register.

4.22 Board Temperature Register (CCD_TEMP)

The board ambient temperature may be accessed by reading the `CCD_TEMP` register. The temperature is expressed as a signed 10-bit number in °C multiplied by 4. For example, 25°C is 100 or 0x064 in hex. Refer to the AD7814 datasheet for more information.

The temperature sensor is *not* read automatically. To get the current temperature the user must first write to the `CCD_TEMP` register to trigger an update. While the temperature sensor is busy the "Temp Busy" flag is set in the `CCD_STATREG` register.

4.23 Board Control Register (CCD_CTLREG)

The board control register shown in Figure 14 is read/write. If the board is reset this register will default to all zeros.

Figure 14: The board control register.

RST DCM Set this bit to reset the Digital Clock Manager device in the FPGA and force it to re-lock onto the 40MHz reference clock. This bit is self-clearing. Refer to section 2.3.1 for details.

PIPE MODE Set this bit to enable Pipeline Mode transfers.

BRST MODE Set this bit enable the Burst Mode transfers.

Data Select The Data Select bits determine how the pixel ADC values are manipulated. The pixel ADC is 18 bits wide however these control bits allow for a subset of this value to be used. Note that these bits affect the pixel values sent in pipeline and burst modes, as well as the direct read ADC registers `CCD_ADCDATA`. The Data Select modes are defined in Table 3. Note that the power on default is 00.

Table 3: The Data Select bits.

Value	Data Range	Data width and apparent gain
00	`CCD_ADCDATA[15:00]`	16-bit pixel data @ full gain
01	`CCD_ADCDATA[16:01]`	16-bit pixel data @ 1/2 gain
10	`CCD_ADCDATA[17:02]`	16-bit pixel data @ 1/4 gain
11	`CCD_ADCDATA[17:00]`	18-bit pixel data @ full gain

Board Priority The board priority value determines how quickly this board gains control of the pixel data bus in Pipeline and Burst modes. Zero is highest priority and six is the lowest priority in this arbitration scheme. If only one board is present in the crate it should be set priority 0. Refer to section 3.3.3 for information on how pipeline and burst mode transfers work.

Testpoint Channel This register selects which physical channel is connected to the front panel test points. The power on default is channel 0 and channel numbers 0 through 11 are allowed. Refer to Appendix D for the the front panel test point pinout.

4.24 Board Status Register (CCD_STATREG)

Firmware modules report their busy status in the `CCD_STATREG` register. The bits are shown in figure 15:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved					SHUT DOWN	OWD ERR	SEQ BUSY	PIX BUSY	OWD CRC	OWD BUSY	TEMP BUSY	TEL BUSY	DAC BUSY	ADC BUSY	DCM LOCK

Figure 15: The board status register.

DCM LCK	is set to indicate that the Digital Clock Manager is locked. Locked is normal. Refer to section 2.3.1 for details.
ADC BUSY	is set if any of the main ADCs are busy reading out data.
DAC BUSY	is set if the FPGA is writing to any of the Offset or Bias DACs.
TEL BUSY	is set if the Telemetry ADCs are busy reading out.
TEMP BUSY	is set if the FPGA is reading the temperature sensor.
OWD BUSY	is set when communicating with the "one wire" serial number/EEPROM device.
OWD CRC	is set if the DS2433 serial number passes the CRC check.
SEQ BUSY	is set if the FPGA is busy waiting to take control of the Monsoon system pixel bus.
OWD ERR	is set if there is a problem reading from the "one wire" serial number/EEPROM device.
SHUT DOWN	is set if the transition board "hot swap" controller detects a problem with the backplane voltages.

4.25 Board Identity Code Register (CCD_IDENTREG)

Reading from the `CCD_IDENTREG` will return the CCDACQ12 board identifier, 0x0502. Writing to this register will reset the board which will clear most registers but leave RAM buffers intact.

4.26 Firmware Version Register (CCD_FIRMVERS)

Reading from the `CCD_FIRMVERS` register will return the FPGA firmware version number in the lower byte and 0x20 in the upper byte.

Writes to this register used to trigger an erase and reprogram cycle on the FPGA, however this feature was not reliable and sometimes resulted in an unresponsive board. This "reboot" feature has been removed from the firmware; force a reset instead.

Appendix A Memory Map

Table 4: The CCDACQ12 memory map.

Address	Function Name	Access	Description	Page
0x0	CCD_CDSREG	RW32	CDS control register.	14
0x1	CCD_DOPREG	RW32	Digitial output port (dummy) register.	15
0x4	CCD_BURST	W32	Write to initiate burst data xfer to MCB.	15
0x5	CCD_SEQTRIG	RW32	Write to trigger the microsequencer.	15
0x0100-b	CCD_ADCDATA	R	ADC data registers (12).	16
0x0200-b	CCD_ADCCFG	RW	ADC configuration registers (12).	16
0x0210-b	CCD_VOFFSETS	RW	Offset DAC registers (12).	16
0x0220-4f	CCD_HVBIASES	RW	Bias DAC registers (48).	16
0x0250-7f	CCD_TELMODE	RW	Telemetry configuration registers (48).	17
0x0280	CCD_AUXCFGREG	RW	Auxilliary configuration register.	17
0x0300-17	CCD_REDIRECT	RW	Redirect registers (24).	18
0x0340	CCD_XFERCOUNT	RW	Pixel transfer count register.	18
0x0440-6f	CCD_TELEMETRY	RW	Telemetry ADC registers (48).	17
0x1000-3f	CCD_PATMEM	RW	Microsequencer vector buffer (64).	18
0x2000-ff	CCD_NVDAT	RW	Non-Volatile RAM buffer (256).	18
0x2100	CCD_NVCTL	W	Non-Volatile EEPROM control register.	19
0x3000	CCD_CAPCTRL	RW	Backplane capture control register.	19
0x3001	CCD_CAPADDR	R	Backplane capture address register.	19
0x3002	CCD_CAPDAT0	R	Backplane capture data register.	19
0x3003	CCD_CAPDAT1	R	Backplane capture data register.	19
0xc000	CCD_CTC_COUNT	R	Audit register.	19
0xc001	CCD_REQ_COUNT	R	Audit register.	20
0xc002	CCD_XFER_COUNT	R	Audit register.	20
0xfff0	CCD_EVENT_REG	RW	Global event register.	20
0xfff8	CCD_VSUB	RW	Substrate voltage control register.	20
0xfff9	CCD_LEDCTL	RW	LED control register.	20
0xfffa	CCD_SERNUM	R	Serial number register.	20
0xfffb	CCD_TEMP	RW	Temperature register.	20
0xfffc	CCD_CTLREG	RW	Board control register.	21
0xfffd	CCD_STATREG	R	Board status register.	22
0xfffe	CCD_IDENTREG	RW	Board identity register (reset).	22
0xffff	CCD_FIRMVERS	RW	Firmware version register.	22

Appendix B Telemetry and Bias Channel Assignments

Although the CCDACQ12 board supports 48 bias DAC channels and 48 telemetry channels, not all of these are actually used. Furthermore, some channels are used for other purposes such as controlling the substrate voltage and monitoring temperatures as shown in Table 5.

Table 5: Telemetry and Bias Channels.

Channel Number(s)	DAC Description	Telemetry ADC Description
0	Vsub upper limit	Vsub upper limit
1	Vsub ramp rate	Vsub ramp rate
2	not used	not used
3	not used	not used
4	no DAC installed	input grounded
5	no DAC installed	input grounded
6	no DAC installed	input grounded
7	no DAC installed	input grounded
8	no DAC installed	RTD 1
9	no DAC installed	RTD 2
10	no DAC installed	RTD 3
11	no DAC installed	RTD 4
12	no DAC installed	RTD 5
13	no DAC installed	RTD 6
14	no DAC installed	MAX1270 Reference 4.096V
15	no DAC installed	Buffered Reference
16-23	Vru	Vru
24-31	Vrl	Vrl
31-39	Vog	Vog
40-47	Vdd	Vdd

Appendix C Calibration Constants

When writing to Offset and Bias DACs and reading from Telemetry ADCs it is useful to use real-world voltage levels and not 12-bit binary values. Integer to floating point conversions are handled in the Monsoon software. The conversion routines assume that the algorithm is linear and thus two parameters or calibration constants are required: slope and offset.

The CCDACQ12 board provides 256 16-bit memory locations which are used to store slope and offset constants for each DAC and Telemetry ADC. These constants may be read or changed at any time – however an additional step is needed to save these constants back to the on-board EEPROM memory, refer to section 4.15 for details.

Each calibration constant is stored as a 16-bit *fixed point* integer as shown in Figure 16. Each constant has a unsigned 14-bit integer portion and a 2-bit fractional component. Values range from 0.00 (0x0000) to 16383.75 (0xFFFF).

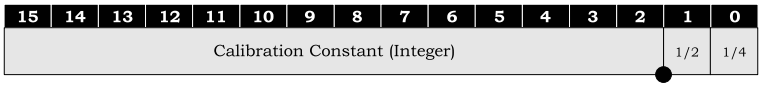

Figure 16: Fixed point calibration constant.

Table 6: Calibration constant memory map (NV_DAT)

NV_DAT Address (dec)	Constant
00	Offset 0 DAC slope
01	Offset 0 DAC offset
:	:
22	Offset 11 DAC slope
23	Offset 11 DAC offset
24	Vbias 0 DAC slope
25	Vbias 0 DAC offset
:	:
118	Vbias 47 DAC slope
119	Vbias 47 DAC offset
120	Telemetry 0 ADC slope
121	Telemetry 0 ADC offset
:	:
214	Telemetry 47 ADC slope
215	Telemetry 47 ADC offset
216-255	Reserved

Appendix D Front Panel Connectors

The front panel test points connect directly to the FPGA and are reserved for debugging signals. The pinouts for the front panel test point connector and the JTAG connector are shown in Figure 17. The names next to the test point connector pins reference FPGA pin numbers.

The ADC and CDS switch signals are brought out to the front panel test points. By default these signals map to channel zero but the target channel may be changed by writing to the upper four bits of the board control register `CCD_CTLREG`. The remaining front panel testpoints are not defined in the production firmware and may be used for adding probes in the Xilinx FPGA Editor tool.

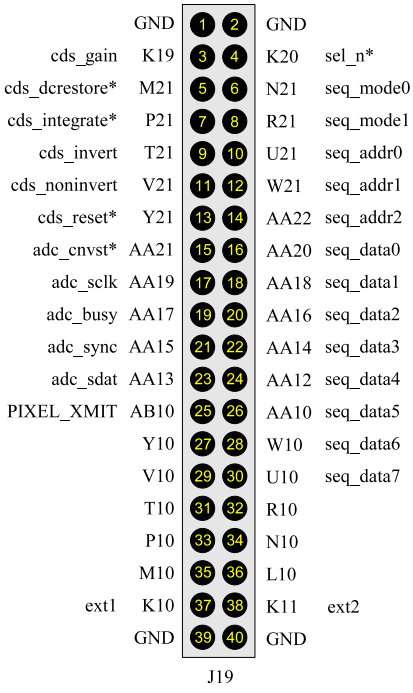

Figure 17: Front panel test points.

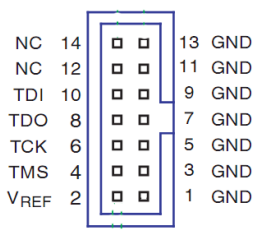

Figure 18: The high speed Xilinx JTAG connector used on version 2 boards.

Appendix E Building the Firmware

E.1 Synthesis

Synthesis is the process of converting the VHDL source code into logic primitives (gates, flop-flops, RAMs, etc.). The "synth tool" we use for this is XST, which is bundled with the Xilinx ISE software suite. If successful the "synth tool" will produce a flattened binary netlist.

The EEPROM module includes a small 8-bit embedded processor ("PicoBlaze") to handle the fairly complicated handshaking over the "one wire" bus. The software that runs on this processor (`PROG2433.PSM`) is converted into a VHDL source file using the `KCPSM3.EXE` assembler, see Xilinx Application note 213 for more information.

E.2 Place and Route

The process of fitting the netlist into the physical device is commonly called "place and route" but in reality it is a collection of Xilinx programs which map the netlist into the FPGA architecture (e.g. individual gates to LUTs), place the components, route the interconnections between components and evaluate timing. These programs are ngdbuild, map, par, trce and bitgen.

E.3 Scripting

The entire build process is controlled not from the Xilinx GUI but from the command line using a makefile. In order to execute the commands in the makefile the utility `make` must be installed on the system. Every Unix/Linux system includes the `make` utility. Windows users must install Cygwin which includes `make` by default.

Firmware releases are zipped archives that contain two directories: `src` and `xilinx`. Unzip the latest archive in any empty directory and `cd` to the `xilinx` directory. Connect the Xilinx programmer cable to the CCDACQ12 board and type `make program` at the command prompt. This will start the Xilinx IMPACT program which will erase the PROM and then download and verify the firmware.

For any reason the user may want to rebuild the firmware from source. To do this stay in the `xilinx` directory and type the following commands: `make clean` followed by `make`. The process should take less than 10 minutes on any modern workstation. After the process is done look at the *.par file and check that the final timing score is zero, which means that the design met all timing constraints. If everything looks good type `make program` to download the configuration file to the board.

E.4 Firmware Source Tree

```
./src
   |-- ccdacq_package.vhd
   |-- fpga_top.vhd
   |-- /adc
   |     +-- adc_interface.vhd
   |-- /backplane
   |     |-- backplane_interface.vhd
   |     |-- pix_driver.vhd
   |     |-- collector.vhd
   |     +-- capture.vhd
   |-- /seq
   |     |-- seq_top.vhd
   |     |-- sequencer.vhd
   |     |-- ram64xwd.vhd
   |     |-- ram64x1d.vhd
   |     +-- cds.vhd
   |-- /dac
   |     |-- dac_interface.vhd
   |     |-- dac_mod.vhd
   |     +-- fifo64.vhd
   |-- /misc
   |     |-- reset_interface.vhd
   |     |-- clock_multiply.vhd
   |     |-- dcm_wrapper.vhd
   |     |-- led_interface.vhd
   |     +-- temp_interface.vhd
   |-- /eeprom
   |     |-- eeprom.vhd
   |     |-- ram256x16.vhd
   |     |-- kcpsm3.vhd
   |     |-- prog2433.vhd
   +-- /tel
         |-- telemetry_interface.vhd
         |-- ram16xnd.vhd
         +-- telem16a.vhd
```

Figure 19: The firmware source tree.

Appendix F Power Requirements

Note that a switching regulator is used to generate the +1.2V FPGA core voltage from the +3.3V rail.

Table 7: Power supplies.

Voltage	Current	Power
+3.3Vd	0.186 A	0.614 W
+5Vd	0.164 A	0.820 W
+5Va	0.875 A	4.375 W
-5Va	0.635 A	3.175 W
+15Va	0.770 A	11.55 W
-15Va	0.493 A	7.395 W
-28V	0.187 A	5.236 W

References

[1] Fermi National Accelerator Laboratory
Batavia, Illinois 60510 USA
http://www.fnal.gov

[2] The Dark Energy Survey
https://www.darkenergysurvey.org

[3] National Optical Astronomy Observatory
Tucson, Arizona 85719 USA
http://www.noao.edu

[4] Moore, Peter
Monsoon 8 Channel CCD Acquisition Board (CCDACQ)
NOAO Document MNSN-AD-08-0004
http://www.noao.edu/ets/monsoon

[5] Xilinx Semiconductor
http://www.xilinx.com

[6] Maxim Semiconductor
http://www.maximic.com

[7] Analog Devices
http://www.analog.com

[8] CCDACQ12 Schematic V2
DES Document 690
http://des-docdb.fnal.gov:8080/cgi-bin/ShowDocument?docid=690